НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «»
спеціальності 123 «Комп'ютерні системи та мережі»
на тему: «Веб-сервіс з автоспортивними новинами»

Виконав (-ла):
студент (-ка) IV курсу, групи ІО-61
Дем'яненко Максим Олександрович _____

Керівник:
Доцент кафедри ОТ, к.т.н.,
Верба Олекснадр Андрійович _____

Консультант з нормоконтролю:
Професор кафедри ОТ, д.т.н.,
Сімоненко Валерій Павлович _____

Рецензент:
Доцент кафедри ПЗКС, к.т.н,
Юрчишин Василь Якович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент_____

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський) Спеціальність – 123 «Комп’ютерна інженерія» Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри

Сергій Стіренко
"___" _____ 2020 року

ЗАВДАННЯ
на дипломний проєкт студенту

Дем’яненку Максиму Олександровичу

1. Тема проекту «Веб-сервіс з автоспортивними новинами» керівник проекту Верба Олександр Андрійович, затверджені наказом по університету від "07" травня 2020 року № 1081-с.
2. Термін подання студентом проєкту 26 травня 2020р.
3. Вихідні дані до проєкту *див. технічне завдавння* .
4. Зміст пояснювальної записки *Аналіз і характеристика об'єкта проектування, обґрунтування оптимального варіанта реалізації мети цієї роботи, розробка додатку: вибір технологій та їх обґрунтування, основні рішення з реалізації додатку. Висновки.*
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) *принципова схема, функціональна схема, структурна схема.*

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормконтроль	Сімоненко В.П., професор, д.т.н.		

7. Дата видачі завдання *01.09.2019.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	*Затвердження теми роботи*	*01.09.2019*	
2	*Вивчення та аналіз завдання*	*15.12.2019-15.03.2020*	
3	*Розробка архітектури додатку*	*15.03.2020-25.03.2020*	
4	*Написання програмної частини*	*25.03.2020-05.04.2020*	
5	*Тестування та виправлення помилок*	*05.04.2020-15.04.2020*	
6	*Оформлення пояснювальної записки*	*15.04.2020-20.05.2020*	
7	*Захист програмного продукту*	*25.04.2020*	
8	*Передзахист*	*26.05.2020*	
9	*Захист*	*17.06.2020*	

Студент **Дем’яненко Максим**

Керівник **Олександр Верба**

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ІАЛЦ.467200.000 ДП	Відомість дипломного проекту	1	
3	А4	ІАЛЦ.467200.002 ТЗ	Технічне завдання	2	
4	А4	ІАЛЦ.467200.004 ПЗ	Пояснювальна записка	50	
5	А4	ІАЛЦ.467200.005 Д1	Схема програми	1	
6	А1	ІАЛЦ.467200.006 Д2	Діаграма класів	1	
7	А1	ІАЛЦ.467200.007 Д3	Схема взаємодії програми з клієнтом	1	

				ІАЛЦ.467200.000 ДП		
	ПІБ	Підп.	Дата			
Розробн.	Дем'яненкоМ.О.			Відомість дипломного проєкту	Лист	Листів
Керівн.	Верба О.А.				1	1
Консульт.					КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІО-61	
Н/контр.	Сімоненко В.П.					
Зав.каф.						

Технічне завдання до дипломного проекту

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується на розробку веб-сервісу з автоспортивними новинами.

Область застосування: організації та компанії.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання розробки веб-сервісу з автоспортивними новинами, затвердженою кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка веб-сервісу з автомантичною обробкою даних з різних автоспортивних джерел.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література з комп’ютерних технологій, довідники з програмовамування, публікації в Інтернеті за даним питанням.

Пояснювальна записка
до дипломного проєкту
на тему: «Веб-сервіс з автоспортивними новинами»

Київ – 2020 року

ЗМІСТ

ВСТУП

На жаль, в Україні авто- та мотоспорт знаходиться в досить поганому стані. За рік відбувається невелика кількість заходів. Деякі види зовсім не розвинуті у зв'язку з відсутністю тих чи інших умов для проведення.

На мою думку, не має кращих способів популяризації будь-якої теми, ніж засоби масової інформації. Українських сайтів, які б освітлювали дану тему або не існує, або вони відомі лише вузькому колу людей. Інші засоби інформації навіть не освітлюють досягнення вітчизняних спортсменів.

Створення будь-якого інтернет сайту або сервісу – це досить важкий процес, який включає в себе розробку фронтенд і бекенд частин. Бекенд відповідає за реалізацію усієї логіки сервісу, безперебійну роботу з базою даних тощо. До фронтенд частини відноситься створення інтерфейсу користувача та функціоналу, що працюють на клієнтській частині веб-сервісу. Головна задача – пов'зати обидві частини так, щоб вони утворювали єдину цілісну ситсему. Інтсрументи та засоби для розробки веб-сайтів постійно розвиваються. З'являється новий функціонал у вже існуючих рішень. Також виходять зовсім нові засоби такі, як бібліотеки, створені користувачами, або окремі мови програмування, що дає програмістам широкий асортимент для втілення будь-яких задумів.

Метою мого бакалаврського проекту є розробка веб-сервісу, який націлений на популяризацію автоспорту в Україні, а це гарна нагода закріпити та поглибити знання, отримані протягом чотирьох років навчання, та використати на практиці навички розробки програмного забезпечення, роботи з базами даних та інше.

РОЗДІЛ 1
АНАЛІЗ АРХІТЕКТУРНИХ ЕЛЕМЕНТІВ ВЕБ-САЙТІВ

1.1 Загальні відомості про веб-сайти. Класифікація та аналіз різних типів веб-сайтів

Сайт, або веб-сайт – одна або декілько логічно зв'язаних між собою вебсторінок, які об'єднанані між собою за змістом та за навігацією під єдинним доменом. Найчастіше сайт являє собою масив зв'язаних між собою даних, що має унікальну адресу і сприймається користувачем, як єдине ціле. Доступ до сайтів відбувається через протокол передачы даних HTTP (англ. *HyperText Transfer Protocol* — «протокол передачи гіпертексту»).

Веб-сайт, як система електронних документів може належати як приватній особі, так і організації, а також бути доступним в комп'ютерній мережі під доменним ім'ям та IP-адрессою або локально на одному комп'ютері. Доменне ім'я являє собою символьне ім'я, що слугує для ідентифікації областей, які є частинами простору ієрархічних імен мережі Інтернет, що обслуговується групою DNS-серверів (серверів системи доменних імен) та адмініструється централізовано. DNS-сервери зберігають інформацію про вузли, імена яких належать домену і виконують трансляцію їх імен в адреси. Сукупність усіх сайтів являє собою Всесвітню мережу, де комунікація об'єднує сегменти інформації всесвітньої спільноти в єдине ціле – базу даних та комунікації планетарних маштабів. Для прямого доступу клієнтів до таких сайтів був розроблений протокол HTTP.

HTTP – протокол прикладного рівня (взаємодія мережі та користувача) передачі даних у вигляді гіпертекстових документів у форматі HTML. Осно-

вою протоколу є технологія «клієнт-сервер», тобто передбачається існування двох сторін:

- споживачів або клієнтів, які ініціюють з'єднання та відсилають запит;
- постачальник або сервер, який чекає з'єднання для отриманні запиту, виконує необхідні дії і повертає повідомлення з результатом.

Сторінки сайтів – набір текстових файлів, розмічений на мові HTML. Ці файли, будучи завантаженими на комп'ютер користувача, розпізнаються і оброблюються браузером і виводяться на на засіб відображення (монітор, екран телефону тощо). Мова HTML дозволяє форатувати текст, виділяти в ньому функціональні елементи, створювати гіпертекстові посилання і вставляти на відображену сторінку зображення, відео- та аудіозаписи та будь-які інші мультимедійні елементи. Відображення сторінки змінюється за допомогою додавання стилів, написаних на мові CSS, що дозволяє змінити розмір, колір блоків, шрифт тексту та інше; а також шляхом додавання сценаріїв створених на мові JavaScript.

Сайти можуть мати підрозділи, орієнтовані на ту чи іншу аудиторію. У такому випадку такі розділи називають версіями сайту. Аудиторія може розрізнятися по виду використовуваного обладнання або мови використання. На приклад, існують так звані мобільні версії сайтів, які призначені для користування за допомогою різних портативних пристроїв. Такі версії потребують або гнучку верстку сайту, яка налаштовується під під розміри екрану, вікна браузера, типу пристроя, на якому використовується сайт, або зовсім іншого дизайну, створеного окремо для мобільної версії. Такі заходи дозволяють полегшити роботу користувача з сайтом незалежно від типу пристрою.

Далі слід розглянути класифікацію інтернет-ресурсів. Сайти розрізняються:

1) За доступністю сервісів:

- Відкриті – всі сервіси цілком доступні для будь-яких відвідувачів;
- Частково відкриті – для доступу необхідно зареєструватися;
- Закриті – цілком закриті службові сайти організацій (в тому числі корпоративні сайти) та особисті сайти приватних осіб. До таких сайтів мають доступ лише користувачі певної групи. Доступ новим користувачам надається через так звані запрошення.

2) За фізичним розташуванням:
   - Загально доступні сайти мережі Інтернет;
   - Локальні сайти – доступні лише в межах локальної мережі.
3) За схемою представлення інформації, її обсягу та категорії вирішуваних задач:
   - Інформаційні ресурси: тематичний сайт (сайт, що надає специфічну вузьконаправлену інформацію з будь-якої теми), тематичний портал (досить великий веб-ресурс, який надає вичерпну інформацію певної тематики. Портали схожі на тематичні сайти, однак додатково мають засоби взаємодії з користувачами і дозволяють користувачам спілкуватися в межах порталу (форуми, чати).
   - Інтернет-портал – багатокомпонентна розгалужена структура, скомпонована ыз самодостатніх сайтів самостійних організацій або підрозділів корпоративної структури.
   - Інтернет-представнитства власників бізнесів.
   - Веб-сервіси.

Інтернет представнитсва в свою чергу поділяються на:

- Сайт-візитка – включає в себе загальні дані про власника сайту: вид діяльності, історія, прейскурант, контактні дані, реквізити. Спеціалісти розміщують своє резюме.

- Представницький сайт – сайт-візитка з розширеним функціоналом: докладний опис послуг, відгуки, портфоліо, форма зворотнього зв’язку тощо.
- Каталог продукції - в каталозі міститься докладний опис товарів чи послуг, сертифікати, технічні та споживчі дані, відгуки експертів і так далі. На таких сайтах розміщується інформація про товари або послуги, яку неможливо помістити в прейскурант.
- Інтернет-магазин - сайт з каталогом продукції. Використовуючи такий сайт клієнт має можливість замовити необхідні йому товари. Використовуються різноманітні системи розрахунків, такі як: пересилання товарів післяплатою чи автоматичною пересилання рахунку по факсу або розрахунків за допомогою пластикових карт.
- Промосайт - сайт про певну торгову марку чи продукт. Такий сайт містить вичерпну інформацію про бренд, різні рекламні акції, конкурси, вікторини, ігри тощо.
- Корпоративний сайт - містить усю інформацію про компанію-власника, її продукцію, послуги та події в житті компанії. Від сайту-візитки і представницького сайту відрізняється обсягом наданої інформації. Досить часто такий сайт містить різні функціональні інструменти для роботи з контентом, а саме: пошук, фільтри, календарі подій, фото- та відеогалереї, корпоративні блоги, форуми тощо. Може бути об’єдгагий з внутрішніми інформаційними системами компанії-власника, наприклад: КІС, CRM, бухгалтерськими системами. Також може містити закриті розділи для певних груп користувачів: співробітників, дилерів, контрагентів тощо.
- Сайт-квест - інтернет-ресурс, на якому організовано змагання з розгадування послідовно взаємопов'язаних логічних загадок.

Веб-сервіси розглянемо в окремому розділі.

За технологічними особливостями створення і відображення сайти розрізняються за:

1) Типами макетів:
    - Фіксованої ширини (англ. Rigid fixed). В таких макетах розміри елементів мають фіксовані значення, які не залежать від розміру, співвідношення сторін екрану та розмірів вікна браузера. Задається в абсолютних значеннях - px (пікселі).
    - Динамічно еластичний тип (англ. Dynamically expandable elastic). Такі макети мають елементи, розміри більшості з яких задаються відносними значеннями, тобто em і% (відсотки). Усі пропорції розмірів елементів залишаються незмінними, незалежно від роздільної здатності, розміру, співвідношення сторін екрану монітора, а також розмірів та масштабу вікна користувача.
    - Гумовий (англ. Adaptable fluid) – розміри головних або несучих елементів, значення ширини, задаються значенням у відсотках. Сторінки відображаються на весь екран по ширині.
    - Адаптивний (англ. Adaptive). У таких макетах дизайн сторінки адаптується під розмір екрану, а також може відбуватися перебудова блоків з одного місця на інше, або заміна одних блоків на інші, при виконанні певних умов.
2) Технологією відображення:
    - Статичні. Такі сайти складаються зі статичних html-сторінок, які утворюють одне ціле. Користувач отримує файли в тому вигляді, в якому вони зберігаються на сервері.
    - Динамічні - складаються з динамічних сторінок-шаблонів, скриптів, інформації тощо у вигляді окремих файлів. Вміст сторінки генерується спеціальними скриптами (програмами) на основі інших даних з будь-якого джерела.

- Сайти, створені із застосуванням Flash-технологій. Весь сайт розташовується на одній веб-сторінці, яка необхідна лише для завантаження Flash-файлу, а вся навігація і вміст реалізовані в самому Flash-ролику.

1.2 Веб-сервіси. Загальні положення. Переваги і недоліки.

Темою даної дипломної роботи є веб-сервіс. Тому слід розглянути даний тип інтернет-ресурсів окремо від інших.

Веб-сервіс або веб-служба – сайт, розроблений для виконання будь-яких задач або надання послуг в межах Всесвітньої мережі. Ідентифікується унікальною веб-адресою (URL-адресою) програмна система зі стандартизованими інтерфейсами, а також HTML-документ сайта, що відображається браузером користувача. Веб-служба могут взаємодіяти одна з одною та сторонніми додатками шляхом повідомлень, які базуються на певних протоколах(SOAP, XML-RPC тощо) та станів(REST).

Веб-сервіси поділяються на велику кількість підтипів, таких як:

- Пошукові сервіси – алгоритми та їх сукупність, що дає користувачу здатність швидкого доступу до необхідної інформації за допомогою пошуку серед обширної колекції доступних даних. Приклади: Google, Yandex, Yahoo!, Bing.
- Поштові сервіси – сервіси які дозволяють користувачам комп’ютерної мережі обмінюватись повідомленнями. Приклади: ukr.net, i.ua, Gmail.
- Веб-форуми – сервіси для спілкування декількох користувачів на одну чи більше тем.
- Дошка оголошень – сервіс з можливістю розташування публічних обьяв вільного характеру.

- Каталог сайтів – ресурс, на якому розміщуються сайти та блоги. Приклад Open Directory Project.
- Блоговий сервіс – веб-сайт, вмістом якого є регулярні записи однієї людини або групи людей.
- Хмарне сховище даних – сервіс, який дозволяє зберігати дані на численних розподілених у мережі серверах, виконувати над ними операції, надавати доступ іншим користувачам тощо. Приклад: Google Drive.
- Сервіс редагування даних. Приклад: Google Docs.
- Фото- та відеохостінги.

Також існують комбіновані веб-сервіси такі, як соціальні мережі. Вони являють собою сукупність веб-сервісів, що дозволяють користувачам спілкуватись між собою, знайомитись, створювати соціальні відносини та утворювати групи.

Будь-яка система має свої переваги та недоліки. Тому звернемо увагу на плюси та мінуси веб-сервісів. До переваг можна віднести те, що веб-служби забезпечують взаємодію програмних систем незалежно від платформи. Також веб-сервіси засновані на базі відкритих стандартів. Завдяки використанню XML досягається простота розробки й відладки веб-служб. Ще одною перевагою є використання інтернет-протоколу забеспечую HTTP-взаємодію програмних систем через міжмережевий екран. Це значний плюс в порівнянні з такими технологіями, як CORBA, DCOM та Java RMI. На інший погляд, веб-сервіси не прив’язані к HTTP і можуть використовувати інші протоколи. Що стосується недоліків, то веб-служби мають меншу продуктивність і більший розмір використовуваного трафіку у порівнянні з технологіями DCOM, CORBA, Java RMI за рахунок використання XML-повідомлень. Однак на деяких веб-серверах є можливість зжимання мереживного трафіку. Також до недоліків слід віднести безпеку, тобто відповідальні за це служби

повинні використовувати кодування або потребувати аутентифікації користувача.

З чого складається веб-сервіс? Розглянемо архітекртуру веб-служб.

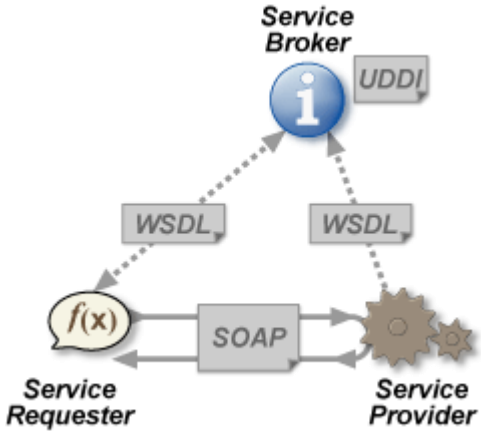


Рисунок 1.1 – Веб-служба на основі UDDI і SOAP.

Розглянувши рисунок 1.1, можна виділити три інстанції, які взаємодіють в межах веб-сервісу: виконавець (service provider), замовник (service requester) та каталог (service broker).

Коли служба готова, виконавець реєструє її в каталозі, де її можуть знайти потенційні замовники. Замовник, знайшовши в каталозі відповідну службу, імпортує звідти її WSDL-специфікацію і розробляє відповідно до неї своє програмне забезпечення. WSDL описує формат запитів і відповідей, якими обмінюються виконавець і замовник в процесі роботи. Для взаємодії використовуються такі стандарти:

- XML: Розширювана мова розмітки, призначена для передачі та зберігання даних певної структури;
- WSDL: Мова визначення зовнішніх інтерфейсів, який визначає взаємодію (контракт) між споживачем і веб-службами SOAP. Написаний на базі XML;
- SOAP: Протокол обміну повідомлень на базі XML;

- UDDI: Універсальний інтерфейс розпізнавання, опису та інтеграції (Universal Discovery, Description and Integration). Каталог веб-сервісів та відомостей про компанії, які надають веб-служби в загальне користування або певним компаніям. Поки що UDDI існують, однак, тільки в невеликих фірмових мережах і ще не знайшли широкого поширення у відкритому інтернеті;
- JSON: Більш ефективний мову розмітки, що став масовим в 2010х роках.

Для даної роботи будемо використовувати саме стандарт JSON, у зв'язку з його розповсюдженністю та простотою використання.

1.3 Платформи для розгортки веб-сервісів. Огляд деяких з них

Веб-служби розгортаються на серверах додатків. Деякі сервери додатків:

- ColdFusion від Adobe;
- DotGNU від GNU Project (розробка зупинена);
- GlassFish — від компанії Oracle;
- Google App Engine — платформа для маштабованих додатків, що використовують інфраструктуру компанії Google;
- IBM Lotus Notes лінійка програмгного забезпечення для організації спільної роботи від IBM;
- JBoss — компанії Red Hat;
- Mono — платформа розробки від Xamarin (раніше Novell);
- .NET Framework сервери від Microsoft;
- Web Application Server від SAP (є ключовою частиною стека SAP NetWeaver)
- WebLogic від компанії Oracle (продукт BEA Systems поглинутої Oracle)
- webMethods Integration Platform від Software AG

- WebSphere Application Server від IBM (заснований на Apache та платформі J2EE)
- Zend Framework — від Zend Technologies.

Висновок до Розділу 1

Таким чином в першому розділі наведені теоритичні відомості про різні види інтернет-ресурсів, їх призначення, їх архітектура. Наведена їх класифікація. Окремо розглянуті веб-служби. Наведені загальні положення про веб-служби, розглянуті їх підтипи з короткою характеристикою кожного з них. Завдяки проаналізованим особливостям були виділені головні недоліки та переваги таких систем, їх орієнтованість на певні групи користувачів. Приведена у першому розділі архітектура дає представлення про склад сучасних веб-сервісів.

РОЗДІЛ 2.

ПРОЕКТУВАННЯ ВЕБ-СЕРВІСУ З АВТОСПОРТИВНИМИ НОВИНАМИ.

2.1 Визначення вимог і завдання. Опис функціоналу сервісу

До основних функцій створюваної системи можна віднести:

- Автоматизоване отримання даних з обраних джерел;
- Обробка отриманих даних та зведення їх до єдиного вигляду;
- Занесення перетворених даних до бази для подальшого використання;
- Відображення даних на клієнтській частині.

Також слід реалізувати віддачу даних з попередньою обробкою на клієнтську частину за мінімальну кількість часу. Інтерфейс користувача повинен бути інтуітивно зрозумілим, мати лаконічний вигляд. Слід уникнути дуже яскравих кольорів, щоб не «різати» око користувачеві. Перехід між розділами повинен також займати невелику кількість часу для збереження максимальної швидкодії сайту.

2.2 Порівняльний аналіз технологій для розробки веб-сервісів

2.2.1 Порівняння Node.js та Python

Кожен проект має певні особливості та вимоги. При створенні програми важливо вибрати правильну технологію. У даному розділі розглянемо особливості Python і Node.js, щоб визначити, яка з платформ найкраще підійде для використання в проекті.

Розглянемо особливості мови Python. Як описано у статті [12] У Python є безліч переваг, що спрощують розробку проектів, від стартапів до великих корпоративних платформ. Ось деякі з них:

- Python скорочує час виходу на ринок. Використовуючи Python програміст може розробляти MVP або прототип в обмежені терміни для швидкого виходу на ринок (TTM). Це досягається завдяки методу швидкої розробки Python. Цей метод дозволяє підтримувати декілька ітерацій одночасно. Також використання принципу DRY передбачає можливість повторного використання частин коду.

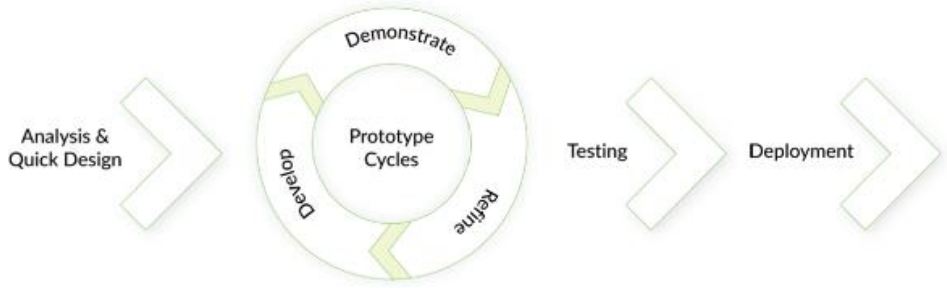


Рисунок 2.1 - Методологія швидкого розвитку додатків

- Простота синтаксису Python. Одна з причин вибору розробниками мови Python полягає в легкому для освоєння синтаксисі, за допомогою якого програміст має змогу висловити концепції в декількох рядках коду, що полегшує пошук та вирішення помилок і налагодження. Він також досить простий для розуміння клієнтами, що дає змогу читати код навіть людині, яка не пов’язана з програмуванням, що полегшує співпрацю.

- Великий набір різноманітних інструментів розробки і фреймворків Python. Потужні веб-фреймворки спрощують процес і дозволяють розробникам зосередитися на логіці додатків.
- Велике співтовариство. У порівнянні з Node.js, Python є більш зрілою open-source мовою і володіє однією з найбільших спільнот, розробники яких постійно діляться рішеннями і покращують мову.

Python відмінно підходить для більшості типів проектів, однак володіє декількома обмеженнями:

- Python є однопоточні мовою. Як і будь-яка інтерпретована мова, Python має більш повільну швидкістю виконання в порівнянні з компільованими мовами (такими як C або Swift). Він може бути не кращим вибором для додатків або сервісів з великою кількістю складних обчислень або для будь-якого проектів, в яких швидкість виконання є найбільш важливою задачею.
- Слабкість в мобільних обчисленнях. Python відмінно підходить для розробки серверних і настільних платформ, але він вважається слабким в розробці мобільних додатків.

У порівнянні з Python Node.js має наступні переваги для веб-розробки:

- Node.js забезпечує високу продуктивність. Тобто, розглядаючи швидкодію, Node.js по цьому параметру кращий, ніж Python. Він заснований на CMS Google V8, що робить його придатним для розробки чат-ботів та будь-яких інших програм реального часу.
- Має усі інструменти для full-stack розробки. Для розробки всього проекту знадобиться лише одна команда розробників, які володіють мовою JavaScript, що дозволяє скоротити витрати.

Недоліками Node.js є:

- Вимагає чистої архітектури. Node.js є подієво-орієнтованим середовищем, тому вона може запускати кілька подій одночасно, за умови добре прописаних відносин між ними.

- Не підтримує завдання з високим завантаженням процесора, так як важкі обчислювальні запити блокують обробку інших завдань та уповільнює роботу всього додатку. Тому ця мова не підходить для проектів, заснованих на науці про дані.

Node.js - це потужна технологія для розробки додатків, таких як різноманітні веб-сервіси, ігрові платформи чи форуми. Node.js добре підходить для обробки проектів з великою кількістю одночасних підключень або додатків з високошвидкісним введенням/виведенням, а також таких додатків, як платформи для підвищення продуктивності, платформи електронної комерції і торгові майданчики P2P.

2.2.2 Відмінності SQL та NoSQL баз даних на прикладі MySQL і MongoDb

Розглянемо різні підходи до архітектури. Звернувшись до статті [13] можна побачити, що зазвичай при розробці додатку використовується головне операційне сховище та деякі додаткові сервіси, наприклад, для кешування або повнотекстового пошуку. Зовсім інший підхід до архітектури з використанням різних баз даних - це мікросервіси. У кожного з таких мікросервісів може бути своя база даних, яка краще оптимізована для певних задач цього сервісу. Як приклад можна привести таку ситуацію: основне сховище може бути на MySQL, Redis і Memcache - для кешування, Elastic Search - для пошуку.

Говорячи про основне операційне сховище, ми маємо такий вибір: з одного боку, ми можемо обрати реляційні бази даних на мові SQL; з іншого боку – обираємо нереляційні бази, а далі вже дивимось на підвиди, які краще підходять в даному випадку. Найбільш типовими NoSQL-моделями даних або document, або key value, або wide column бази даних. Приклади: Memcache, MongoDB, Cassandra, відповідно.

Чому в даному випадку ми порівнюємо MySQL і MongoDB? Якщо подивитися на рейтинги баз даних, то можна побачити, що MySQL, - найбільш популярна реляційна база даних, а MongoDB - найбільш популярна з нереляційних база даних. Тому їх розумно порівнювати.

Якщо говорити про MySQL - це перевірена технологія. MySQL використовується великими компаніями більше 15 років. Через те, що MySQL використовує стандарт SQL, є можливість досить простої міграції на інші SQL-бази даних, якщо захочеться. Також є можливість транзакцій та підтримуються складні запити, включаючи аналітику.

Перевага MongoDB у тому, що у ця база має гнучкий JSON-формат документів. Для деяких завдань розробникам це зручніше, ніж витрачати час на додавання колонок в SQL-базах даних. Не потрібно вчити SQL - для деяких це складно. Також прості запити рідко створюють проблеми, і якщо подивитися на проблеми продуктивності, то зазвичай вони виникають тоді, коли люди пишуть складні запити з JOIN в купу таблиць і GROUP BY. Якщо такої функціональності в системі немає, то створити складний запит виходить складніше. У MongoDB вбудована досить проста масштабованість з використанням технології шардінга. Складні запити якщо виникають, то частіше всього вони вирішуються на стороні додатку. Тобто, якщо нам потрібно зробити щось на зразок JOIN, ми можемо вибрати дані, потім вибрати дані по посиланнях і потім їх обробити на стороні додатку, а для багатьох це набагато простіше, ніж розбиратися з JOIN.

Підхід до розробки і життєвий цикл додатків. Якщо говорити про додатки, де використовується MongoDB, і на чому вони фокусуються - це дуже швидка розробка, бо все можна постійно змінювати формат даних, через гнучкий JSON-формат документа. Другий момент - це схема даних. Тут потрібно розуміти, що у даних завжди є схема, питання лише в тому, де вона реалізується. Ви можете реалізовувати схему даних у себе в додатку, тому що

якимось же чином ви ці дані використовуєте. Або ця схема реалізується за допомогою бази даних.

Дуже часто якщо у вас є якийсь додаток, з даними в базі даних працює тільки цією програмою. Наприклад, ми зберігаємо дані з цього додатка в цю базу даних. Схема на рівні додатку працює добре. Якщо у нас одні і ті ж дані використовуються багатьма додатками, то це дуже незручно, складно контролювати. Тут виникає також питання часу життя додатку. З MongoDB добре робити додатки, у яких дуже обмежений цикл життя. Тобто якщо ми робимо додаток, яке живе недовго, наприклад, сайт для запуску фільму або олімпіади. Ми прожили кілька місяців після цього, і це додаток практично не використовується. Якщо додаток живе довше, то тут вже інше питання.

MySQL відноситься до реляційних база даних, тому ми можемо за допомогою легко відображати зв'язки між таблицями. Нормалізуючи дані, ми можемо змушувати зміни даних відбуватися атомарно в одному місці. Коли дані у нас денормалізовани, нам не потрібно при якісь зміні модифікувати купу документів. Добре це чи погано? Результат - завжди таблиця. На перший погляд, це просто, з іншого боку деякі структури даних не завжди добре лягають в таблицю, через що нам може бути незручно з працювати з такими даними. Говорячи про практичне використання MySQL, часто доводиться денормалізувати дані, іноді для деяких додатків потрібно використовувати щось подібне: зберігаємо JSON, XML або іншу структуру в колонках додатка. У MongoDB структура даних заснована на документах, і дані для багатьох веб-додатків дуже просто відображати. Тому що структура даних MongoDB виглядає як асоційований масив додатку. Це дуже просто і зрозуміло для розробника і дані легко серіалізуются в JSON-документ. Розкладати таке в реляційної базі даних за різними табличками - завдання більш нетривіальне. Результатом є список документів, у яких може бути абсолютно різна структура, тобто отримуємо більш гнучке рішення.

Розглянемо продуктивність. Продуктивність дуже складно порівнювати безпосередньо, тому що ми часто робимо різні схеми баз даних, дизайн програми. Але якщо говорити в цілому, MongoDB спочатку була зроблена, щоб добре масштабироваться на багато вузлів через шардінг, тому ефективності було приділено менше уваги.

Якщо розглядати маштабованість, то перш за все слід сказати, що маштабованість – це те, наскільки легко нам взяти наш маленький додаток і масштабувати його на багато мільйонів, можливо, навіть на мільярди користувачів.

Масштабованість буває різною. В рамках однієї машини, коли ми хочемо підтримувати середні за розміром додатки, тобто середня масштабованість або масштабованість на кластері, коли у нас додатки вже дуже великі і зрозуміло, що навіть одна потужна машина не впорається з поставленою задачею. Також має сенс говорити про те, масштабується читання, запис або обсяг даних. В MySQL в нових версіях досить гарна масштабованість в рамках одного вузла для LTP-навантажень. Якщо у нас маленькі транзакції, ми маємо машину, в якій 64 процесора, то вся система масштабується досить добре. Аналітика або складні запити масштабируются погано, тому що MySQL може використовувати для одного запиту тільки один потік, що погано. Традиційно читання в MySQL масштабується з реплікацією, а запис і розмір даних - через шардінг. Якщо дивитися на усі великі компанії, такі як Facebook, Twitter - вони всі використовують шардінг. Традиційно шардінг в MySQL використовується вручну. Є деякі фреймворки для рішення цієї задачі. Наприклад, Vitess - це open source фреймворк, який Google використовує для scaling сервісу YouTube. До цього був framework Jetpants. Стандартного рішення для шардінга MySQL не пропонує, часто перехід на шардінг вимагає уваги від розробників.

MongoDB з самого початку фокусувався на масштабованості на багатьох вузлах. Навіть у випадках з маленьким додатком рекомендується використовувати шардінг з самого початку.

У шардінгу MongoDB є деякі обмеження, а саме: не всі оператори з ним працюють, наприклад, є isolated-варіант для забезпечення консистентності. Вона не працює якщо використовувати шардінг. Але при цьому багато основних операцій добре працюють з шардінгом, тому людям дозволяється scale'-ити додатки значно краще. Тому можна зробити висновок, що шардінг і взагалі реплікація в MongoDB зроблені куди краще, ніж в MySQL.

2.2.3 Порівняння інструментів розробки фронтенду. Angular, React та Vue.js

Розглянемо кожен з фреймвоків, їх особливості, переваги та недоліки. Ознайомившись зі статтею [11], можно зробити деякий аналіз. Почнемо з Angular. Перевагами Angular є:

- Angular завжди використовується разом з Typescript і має винятково підтримку для цього.
- Angular-language-service - забезпечує інтелектуальні можливості і автозаповнення шаблону HTML-компоненту.
- Детальна документація, яка дозволяє розробнику отримати всю необхідну інформацію.
- Одностороння прив'язка даних, яка забезпечує виняткову поведінку додатка, що зводить до мінімуму ризик можливих помилок.
- Впровадження залежностей від компонентів, пов'язаних з модулями.
- Структура і архітектура, спеціально створені для великої масштабованості проекту.

- MVVM (Model-View-ViewModel), яка дозволяє розробникам працювати окремо над одним і тим же розділом програми, використовуючи один і той же набір даних.

Недоліками Angular є:

- Різноманітність різних структур ускладнює вивчення в порівнянні з React і Vue.js, у яких є тільки «Component».
- Відносно повільна продуктивність. З іншого боку, це можна легко вирішити, використовуючи так званий «ChangeDetectionStrategy», який допомагає вручну контролювати процес рендеринга компонентів.

Далі розглянемо особливості React у порівнянні з іншими. Перевагами React є:

- Легко для освоєння, завдяки простому дизайну, використання JSX для шаблонів і докладної документації. Розробники витрачають більше часу на написання сучасного JavaScript і менше турбуються про код, специфічний для фреймворка.
- Дуже швидкий, через реалізацію React Virtual DOM та різним оптимізаціям рендеринга.
- Гарна підтримка рендеринга на стороні сервера, що робить його потужною платформою для контент-орієнтованих додатків.
- Відмінна підтримка Progressive Web App (PWA) завдяки генератору додатків create-react-app.
- Прив'язка даних є односторонньою, що означає менше небажаних побічних ефектів.
- Redux – це, найпопулярніша платформа управління станом додатків в React, її легко вчити і використовувати.
- React реалізує концепції функціонального програмування (FP), створюючи простий в тестуванні і багаторазово використовуваний код.

- Додатки можуть бути створені використовуючи TypeScript або Facebook's Flow, що мають вбудовану підтримку JSX.
- Перехід між версіями, дуже простий, бо Facebook надає «кодові модулі» для автоматизації більшої частини процесу.
- Навички, отримані під чам вивчення React, можуть бути легко застосовані для розробки на React Native.

До недоліків React.js можна віднести:

- React не однозначний і залишає розробникам можливість вибирати кращий спосіб розвитку. Це може бути вирішено сильним лідерством проекту і хорошими процесами.
- Розробники ділиться за способами написання CSS в React: традиційні таблиці стилів (CSS Modules) і CSS-in-JS (тобто Emotion і Styled Components).
- React відходить від компонентів на основі класів, що може стати перешкодою для розробників, яким більш комфортно працювати з об'єктно-орієнтованим програмуванням (ООП).
- Змішування шаблонів з логікою (JSX) може збити з пантелику деяких розробників при перших знайомствах з React.

Перейдемо до Vue.js. Серед переваг можна виділити:

- Посилений HTML. Vue.js має багато характеристик схожих з Angular, а це допомагає оптимізації HTML- блоків, завдяки використанню різних компонентів.
- Vue.js має дуже детальну документацію, яка може прискорити процес навчання для розробників і заощадити багато часу на розробку програми, використовуючи тільки базові знання HTML і JavaScript.
- Адаптивність. Може бути здійснений швидкий перехід від інших фреймворків до Vue.js через схожість з Angular і React з точки зору дизайну і архітектури.

- Vue.js можна використовувати як для створення простих односторінкових додатків, так і для більш складних веб-сервісів. Важливо, що невеликі інтерактивні елементи можна легко встроювати в існуючу інфраструктуру без негативних наслідків.
- Масштабування. Vue.js може допомогти в розробці великих шаблонів багаторазового використання, які можуть бути зроблені майже за той же час, що і більш прості.
- Невеликий розмір. Vue.js займає близько 20 КБ, при цьому зберігає свою швидкість і гнучкість, що дозволяє досягти набагато кращої продуктивності в порівнянні з іншими платформами.

Серед недоліків виділяють:

- Недолік ресурсів. Vue.js займає досить невелику частку ринку в порівнянні з React або Angular. Це означає, що обмін знаннями в цьому середовищі все ще перебуває на початковій стадії. Невеликий процент розробників на Vue.js у порівнянні з іншими платформами.
- Ризик надмірної гнучкості. Іноді у Vue.js можуть виникнути проблеми при інтеграції в величезні проекти.

Висновок до Розділу 2

У другому розділі було описане поставлене завдання та вимоги. На основі отриманих знань були порівняні різні платформи та інструменти для розробки різних складових веб-сервісу. Був проведений аналіз двох головних видів баз даних на прикладі MySQL та MongoDb, виділені їх відмінності, переваги та недоліки. Проаналізовані засоби розробки дали змогу визначити, які інструменти слід використовувати для виконання поставленої задачі та поглибити знання з даної теми.

РОЗДІЛ 3.
РОЗРОБКА ВЕБ-СЕРВІСУ.

3.1 Вибір технологій та інструментів для реалізації задачі

3.1.1 Вибір платформи для розробки та мови програмування

Так як поставлена задача не вимагає точних складних математичних обчислень, а потребує швидке оперування невеликими обсягами даних, мультиплатформенність, було вирішено обрати Node.js.

У якості фреймворку для розробки веб-сервісу було обрано Feathers.js, створений на основі Express.js.

Express.js - фреймворк веб-додатків для Node.js, який створений як вільне і відкрите програмне забезпечення. Express.js розроблений для створення веб-додатків і API (application programming interface). Насправді Express є стандартним каркасом для Node.js. Цей фреймворк мінімалістичний і включає велику кількість додаткових плагінів.

Feathers.js має більш розширений функціонал, у порівнянні з Express. Для Feathers характерний архітектурний стиль REST.

REST (англ. Representational State Transfer - «передача стану уявлення») - архітектурний стиль взаємодії компонентів додатка в мережі. REST має узгоджений набір обмежень, які враховуються під час проектування розподіленої гіпермедіа-системи. У деяких випадках, а саме у пошукові системах, інтернет-магазинах та інших системах, заснованих на даних, це призводить до спрощення архітектури і підвищення продуктивності. У широкому сенсі компоненти в REST взаємодіють на зразок взаємодії клієнтів і серверів у Всесвітній павутині.

Розглянемо властивості архітектури REST:

Властивості архітектури, які залежать від обмежень, накладених на REST-системи:

- Продуктивність. Взаємодія компонентів системи може бути домінуючим фактором продуктивності і ефективності мережі з точки зору користувача;
- Масштабованість. Використовується для забезпечення великої кількості компонентів та їх взаємодії.

Рой Філдінг, який є одним із засновників специфікації протоколу HTTP, описує вплив архітектури REST на масштабованість так:

- Простота уніфікованого інтерфейсу;
- Відкритість компонентів до можливих змін для задоволення мінливих потреб (навіть при працюючому додатку);
- Прозорість зв'язків між компонентами системи для сервісних служб;
- Переносимість компонентів системи шляхом переміщення програмного коду разом з даними;
- Надійність, яка виражається у стійкості до відмов на рівні системи при наявності відмов окремих компонентів, з'єднань або даних.

Існує принаймні шість обов'язкових обмежень для розробки розподілених REST-сервісів по Філдінгу.

Виконання таких вимог є обов'язковим для REST-додатків. Накладені обмеження визначають роботу сервера в тому, як він може обробляти і відповідати на запити клієнтів. Працюючи в рамках цих обмежень, система набуває такі властивості як продуктивність, простота, масштабованість, здатність до змін, відстеження, переносимість та надійність.

Якщо сервіс порушує будь-яку з умов, дану систему можна вважати REST-системою.

Обов'язковими умовами є:

1. Першим обмеженням, які можуть застосовуватися до гібридної моделі, є приведення архітектури до моделі клієнт-сервер. Відділення функцій інтерфейсу клієнта від функцій сервера, який зберігає дані, підвищує переносимість коду клієнтського інтерфейсу на інші платформи, а спрощення серверної частини покращує масштабованість. Найбільше ж вплив на мережу Інтернет, має саме розмежування, яке дозволяє окремим частинам розвиватися незалежно один від одного.
2. Відсутність стану. Протокол взаємодії між клієнтом і сервером вимагає дотримання наступного умови: в проміжку між запитами клієнта жодна інформація про стан клієнта на сервері не зберігається (протокол без збереження стану). Всі запити від клієнта повинні бути складені так, щоб сервер отримував усю потрібну інформацію для виконання запиту. Стан сесії зберігається на стороні клієнта. Інформація про стан сесії може бути передана сервером будь-якому іншому сервісу (наприклад, в службу бази даних) для підтримки стійкого стану, наприклад, на період встановлення аутентифікації. Клієнт розпочинає відправку запитів, коли він готовий або виникає необхідність перейти в новий стан. Під час обробки клієнтських запитів клієнт знаходиться в перехідному стані. Кожен окремий стан додатку представлено зв'язками, які можуть бути задіяні при наступному зверненні клієнта.
3. Кешування. Клієнти, а також проміжні вузли, можуть виконувати кешування відповідей сервера. Відповіді сервера, в свою чергу, повинні мати неявне чи явне позначення як кешованих або некешованих за для запобігання одеожання клієнтами застарілих або невірних даних у відповідь на наступні запити. Правильне використання

кешування здатне повністю або частково усунути деякі клієнт-серверні взаємодії, ще більше підвищуючи продуктивність і масштабованість системи.

4. Одноманітність інтерфейсу. Наявність уніфікованого інтерфейсу є фундаментальним вимогою дизайну REST-сервісів. Уніфіковані інтерфейси дозволяють кожному з сервісів розвиватися незалежно. До таких інтерфейсів пред'являються наступні обмеження:
    - Ідентифікація ресурсів. Всі ресурси ідентифікуються в запитах, наприклад, з використанням URI в інтернет-системах. Ресурси концептуально відокремлені від уявлень, які повертаються клієнтам. Як приклад, сервер може надіслати дані з бази даних у вигляді HTML, XML або JSON, жоден з яких не є типом зберігання всередині сервера.
    - Гіпермедіа як засіб зміни стану програми. Клієнти змінюють стан системи тільки через дії, які динамічно визначені в гіпермедіа на сервері. Прибираючи прості точки входу в сервіс, клієнт не може припустити, що доступна якась операція над якимось ресурсом, якщо не отримав інформацію про це в попередніх запитах до сервера. Не існує універсального формату для надання посилань між ресурсами, RFC 5988 і JSON Hypermedia API Language є двома популярними форматами надання посилань в REST HYPERMEDIA сервісах.
    - Маніпуляція ресурсами через представлення. Якщо клієнт зберігає представлення ресурсу, включаючи метадані, то він володіє достатньою інформацією для зміни або видалення ресурсу.
    - «Самоописуючі» повідомлення. Кожне повідомлення містить достатньо інформації, щоб розуміти, яким чином його обробляти.

5. Слої. В зв'язку з ієрархічною структурою мереж, клієнт зазвичай не здатний точно визначити, взаємодіє він безпосередньо з сервером або ж з проміжним вузлом. Застосування проміжних серверів здатне підвищити масштабованість за рахунок розподіленого кешування і балансування навантаження. Проміжні вузли також можуть підкорятися політиці безпеки з метою забезпечення конфіденційності інформації.
6. Код на вимогу (необов'язкове обмеження). REST може дозволити розширити функціональність клієнта за рахунок завантаження коду з сервера у вигляді аплетів або сценаріїв. Філдінг стверджує, що додаткове обмеження дозволяє проектувати архітектуру, підтримуючу бажану функціональність в загальному випадку, але, можливо, за винятком деяких контекстів.

Філдінг вказував, що додатки, які не відповідають наведеним умовам, не можуть називатися REST-додатками. Якщо ж всі умови дотримані, то, на його думку, додаток отримає наступні переваги:

- Надійність (за рахунок відсутності необхідності зберігати інформацію про стан клієнта, яка може бути втрачена);
- Масштабованість;
- Прозорість системи взаємодії (особливо необхідна для додатків обслуговування мережі);
- Продуктивність (за рахунок використання кеша);
- Легкість внесення змін;
- Простота інтерфейсів;
- Портативність компонентів;
- Здатність еволюціонувати, пристосовуючись до нових вимог (на прикладі Всесвітньої павутини).

У якості бази даних було обрано MongoDb, так як формат JSON може мати будь-яку структуру, легко читається людиною. До того ж Feathers.js має

усі інструменти для взаємодії з цією базою даних. Для зв'язку з колекціями існують сервіси бази даних. Створюються вони досить просто. Достатньо у корні проекту викликати термінал і ввести команду feathers generate service, слідувати інструкціям. Подальше користування колекціями відбувається за допомогою методів find, get, patch, remove, update а також hook-функцій - підключаємі функції проміжного програмного забезпечення, які можуть бути зареєстровані до (before), після (after) або або в разі помилок (error) методу сервісу.

Node.js дозволяє легко використовувати будь-які сторонні бібліотеки за допомогою NPM(Node Package Manager). Для встановлення модуля необхідно у корні проекту ввести команду npm install [назва модуля]. Npm спробує знайти у відомих йому репозиторіях і встановити останню версію (якщо не вказана у команді) модуля у папку node_modules. Після встановлення модуля у файлі package.json автоматично встановлюється назва і версія модуля у полі dependencies. Npm дозволяє не завантажувати усі використовувані сторонні модулі у репозиторій проекту. Для встановлення усіх необхідних для проекту використовується команда npm install без вказування певного проекту. Дана команда встановить усі модулі, вказані в полі dependencies файлу package.json.

Для роботи зі сторонніми ресурсами будемо використовувати наступні модулі:

- request-promise-native – вбудований модуль, що дозволяє робити HTTP запити
- cheerio – модуль, що дозволяє перетворювати дані у форматі HTML у повноцінний DOM-об'єкт використовуючи jQuery селектори.

Також нам необхідно виконувати одні й ті самі функції з певною періодичністю для постійного оновлення даних. Для цього будемо використовувати модуль agenda.

Для розробки фронтенд частини будемо використовувати React, так даний фреймворк має досить високу продуктивність. Його особливостями є:

1. JSX. JavaScript XML (JSX) - це розширення синтаксису JavaScript, яке дозволяє використовувати HTML-подібний синтаксис для опису структури інтерфейсу. Як правило, компоненти написані з використанням JSX, але також є можливість використання звичайного JavaScript.
2. Односпрямована передача даних. Властивості передаються від батьківських компонентів до дочірнім. Компоненти отримують властивості як безліч незмінних (англ. Immutable) значень, тому компонент не може безпосередньо змінювати властивості, але може викликати зміни через callback-функції. Такий механізм називають «властивості вниз, події наверх».
3. Віртуальний DOM. React створює кеш-структуру в пам'яті, що дозволяє обчислювати різницю між попереднім і поточним станами інтерфейсу для оптимального поновлення DOM браузера. Таким чином програміст може працювати зі сторінкою, вважаючи, що вона оновлюється вся, але бібліотека самостійно вирішує, які компоненти сторінки необхідно оновити.
4. Методи життєвого циклу. Дозволяють розробнику запускати код на різних стадіях життєвого циклу компонента. наприклад:
    - shouldComponentUpdate - дозволяє запобігти перерисовку компонента за допомогою повернення false, якщо перерисовка не потрібна.
    - render - найважливіший метод життєвого циклу. Кожен компонент повинен мати цей метод. Зазвичай викликається при зміні даних компонента для перемальовування даних в інтерфейсі.

- componentDidMount - викликається після першої відтворення компонента. Часто використовується для запуску отримання даних з віддаленого джерела через API.

5. React Hooks. Хуки дозволяють використовувати стан і інші можливості React без написання класів. Побудова призначених для користувача хуків дозволяє поміщати логіку компонента в повторно використовувані функції

3.2 Розробка веб-сервісу.

Як було описано в попередніх розділах, розробка веб-сервісу складається з двох головних частин, пов'язаних між собою. Розпочнемо з бекенду нашого сайту.

Так як у якості фреймворку серверної частини було обрано Feathers.js, створення початкового каркасу веб-додатку не займає багато часу, а саме необхідно ввести команду feathers generate app у корні майбутнього проекту і прослідувати за інструкціями, вказуючи назву проекту, тип та інше. Таким чином початкова структура буде створена автоматично. Далі встановлюємо необхідні модулі, які були описані в попередньому розділі. Також для роботи з базою слід створити сервіси. Нам знадобляться три колекції: calendar, news, series, а також ще одна службова, про яку поговоримо пізніше. Колекція calendar буде використовуватися для зберігання найближчих спортивних подій, із яких будемо будувати календар. Для зберігання автоспортивних серій(категорій) буде використовуватись колекція series. Вона необхідна для сортування новин за серіями. Для збереження самих новин слугує колекція news.

Для того щоб новини були актуальними, необхідно регулярно отримувати дані з джерела. Для цього будемо використовувати модуль agenda. Як було описано у попередньому розділі, за допомогою даного модуля, розроб-

ник створює функції, що називаються jobs(далі будемо називати їх «джобами»). Такі джоби дозволяють виконувати будь-яку функцію в певні періоди часу. Саме це нам і потрібно. Для цього модуля необхідна окрема колекція, у яку зберігається інформація про стан джоб. У документах даної колекції зазначається час останнього запуску та завершення, час наступного запуску, інтервал, з яким виконується функція, та час, коли «джоба» була заблокована, якщо сталася така ситуація.

Далі слід приступити до розробки функцій, які будуть брати дані з джерел, перетворювати їх у потрібний нам вигляд JSON-формату, та класти у колекції. Такі функції називають парсерами. У якості джерела будемо використовувати англійські версії сайтів motorsport.com та redbull.com/us-en/tags/motoring. На жаль, української версії даних сайтів не існує, а англійські є найбільш повними. Для цієї задачі будемо використовувати такі модулі, як request-promise-native і cheerio. Перший дозволяє виконувати звичайні HTTP запити, такі як GET, POST та інші. За допомогою цього модуля будемо отримувати необхідні html сторінки. Для використання другого модуля необхідно володіти JQuery селекторами. За допомогою таких селекторів, даний модуль дозволяє знаходити необхідні DOM елементи із завантаженої сторінки, та брати необхідні її атрибути. Так як і календар і новини можна дістати з однієї сторінки, немає сенсу завантажувати одну й ту саму сторінку двічі. Тому достатньо створити одну функцію, для отримання цих даних.

Розглянемо алгоритм цієї функції:

1. Раз на годину завантажуємо сторінку motorsport.com/all/news і перетворюємо її в DOM-об’єкт за допомогою cheerio;
2. Знаходимо усі елементи, що мають клас “ms-item--art”. Саме ці елементи відповідають за відображення новин(Рис. 3.1).

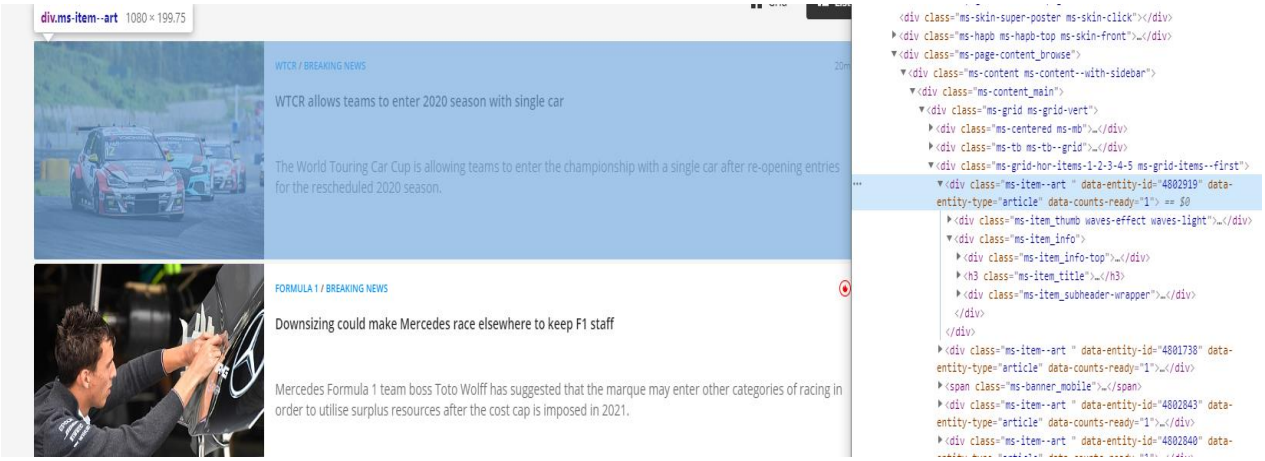


Рис. 3.1. Елемент з класом “ms-item--art”.

3. Проходимо по кожному елементу;
4. Дістаємо заголовок та Url сторінки новини;
5. Шукаємо в базі: якщо новина з таким Url уже існує у колекції news, переходимо до наступної ітерації, якщо не існує – отримуємо картинку, описання, серію;
6. Робимо запит сторінки самої новини, використовуючи її Url;
7. Отримуємо зі сторінки дату новини, автора, головну картинку або галерею картинок, підзаголовок та текст новини;
8. Шукаємо в базі: якщо в колекції series існує документ з назвою серії даної новини, записуємо ідентифікатор серії у поле series новини, інакше – створюємо таку серію і записуємо її ідентифікатор у поле series;
9. Формуємо Url новини для нашого сайту;
10. Якщо ми отримали необхідні дані – створюємо запис новини у колекції news;
11. Отримуємо елементи календара. Елементи календара мають клас “ms-event-strip-item”(Рис. 3.2);

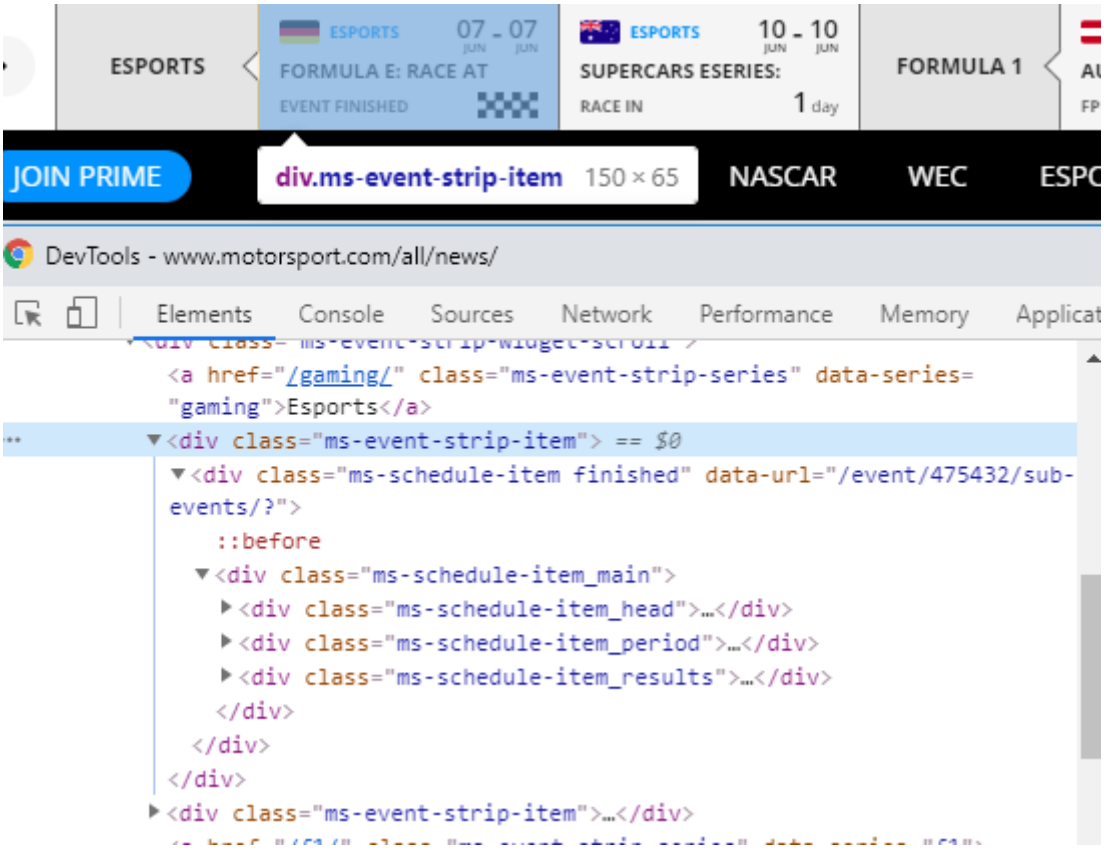


Рис. 3.2. Елемент з класом “ms-event-strip-item”.

12. Проходимо по елементах;
13. Отримуємо назву, серію, дату проведення, та статус події.
14. Шукаємо в базі: якщо в колекції series існує документ з назвою серії даної події, записуємо ідентифікатор серії у поле series події, інакше – створюємо таку серію і записуємо її ідентифікатор у поле series;
15. Шукаємо по назві події у колекції calendar: якщо існує такий запис – оновлюємо, якщо ні – створюємо.

З кодом даної функції можна ознайомитись у додатках.

Для сайту redbull.com будемо виконувати ті ж дії, що й для motor-sport.com, окрім календара, оскільки ми його вже маємо. Алгоритм той самий: завантажуємо сторінку, знаходимо елементи з новинами, перевіряємо на наявність у базі, завантажуємо сторінки кожної з новин, та отримуємо необхідні дані, звівши їх до потрібного нам вігляду. Це буде друга «джоба», яка буде також виконуватись кожної години.

Для забезпечення стабільної роботи сервісу, усі запити необхідно обгортати в оператори try..catch і виводити помилку в консоль, для того щоб бачити, на якому етапі виникають проблеми. Такий підхід дозволяе продовжувати роботу сервісу, а не припиняти його роботу при дрібних помилках. Також слід усі строкові дані обробляти методом .trim(), який прибирає зайві відступи. Якщо цього не зробити, можно потрапити у ситуацію, коли із-за зайвих пробілів або табуляцій створюються дублікати записів у базі даних, через те, що не було знайдено відповідного документу у колекції. Усі дати необхідно тримати в одному форматі, для того щоб можна було виконувати необхідні операції з ними, наприклад нам необхідно буде виводити календар у порядку зростання дати початку події та не виводити ті, які завершились до даного моменту.

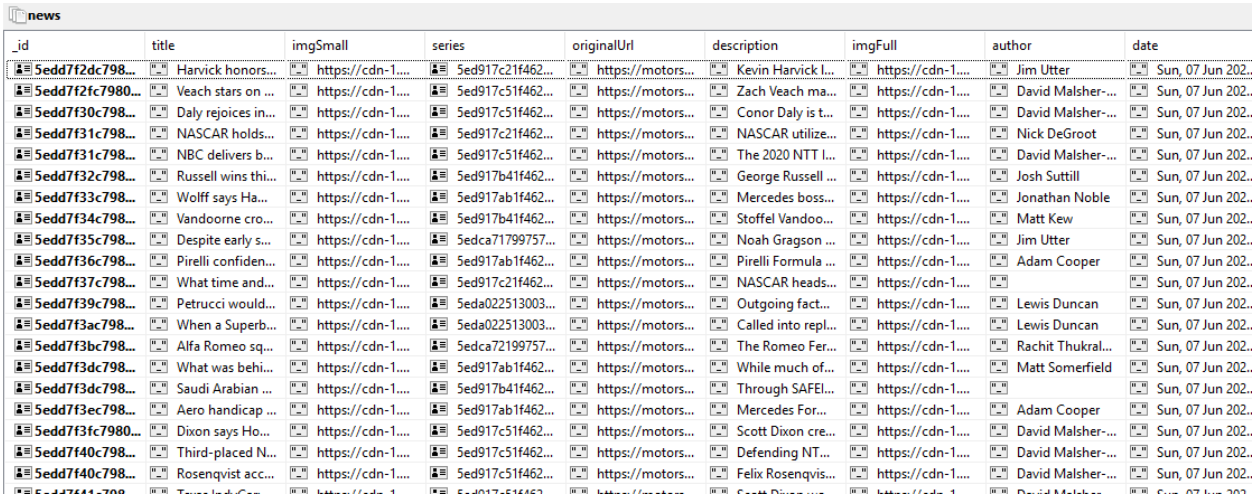
Так як мова JavaScript не має типізації, слід обережно працювати з даними, щоб уникнути помилок та ситуацій, коли, наприклад, функція очікує на вхід об’єкт, а приходить число чи строка.

На даному етапі ми вже маємо базу даних(Рис. 3.3, Рис. 3.4, Рис. 3.5), яка постійно поповнюється, але нам слід перетворити її в повноцінне API. Для цього нам знадобиться такий інструмент, як feathers hooks. Як згадувалось у раніше, кожен сервіс бази даних у feathers має два і більше файли, наприклад: news.service.js, news.hooks.js та інші. У першому файлі описується зв’язок між сервісом та колекцією бази даних, та встановлюється шлях, за яким ми можемо мати доступ до нього. У другому описуюються дії, які будуть виконуватися при тих чи інших запитах до сервісу. Можна вказати, що робобити перед виконанням запиту та після. Для більш зручного користування API розробимо функції для обробки query-параметрів запиту. Запити можна роботи за допомогою звичайних HTTP запитів. Для безпеки можна додати автентифікацію, і перевіряти, чи зареєстрований користувач у системі, і якщо так, то віддавати дані при запиті. Приклад запиту:

GET http://localhost:3030/api/calendar?startDate[$gte]= ISODate("2020-06-08T04:46:42.267+0000")

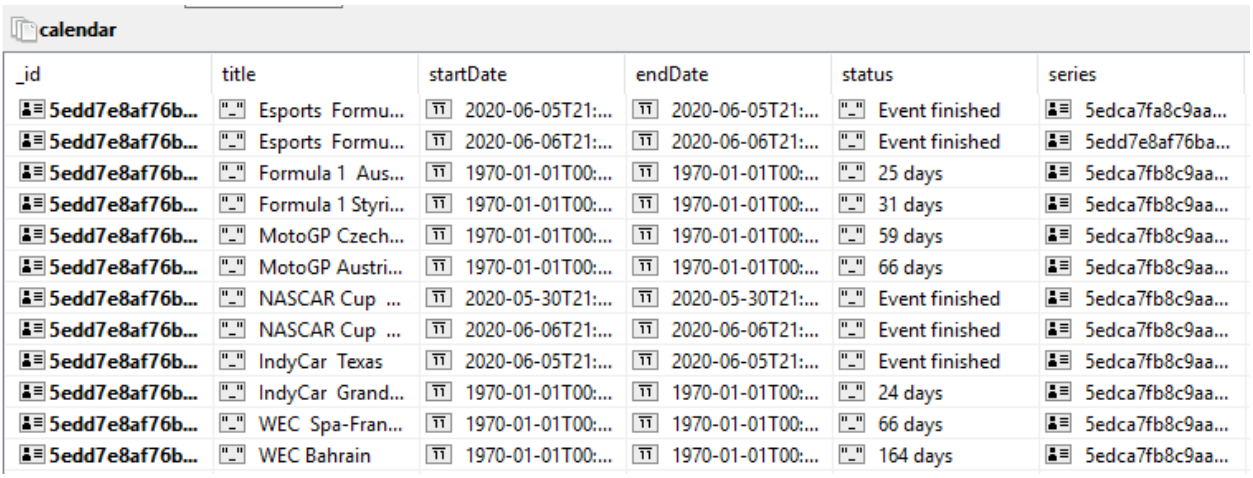
Такий запит повинен повернути усі документи колекції calendar, у яких поле startDate має значення більше, або дорівнює значенню зазначеному у параметрах.

Таким чином була створена серверна частина, яка являє собою API, яке постійно поповнюється даними та оновлює уже існуючі дані. До даного API будемо звертатись, щоб отримати дані у форматі JSON, які будемо використовувати для виводу на клієнті. Далі переходимо до створення фронтенд частини.


news

_id	title	imgSmall	series	originalUrl	description	imgFull	author	date
5edd7f2dc798...	Harvick honors...	https://cdn-1....	5ed917c21f462...	https://motors...	Kevin Harvick I...	https://cdn-1....	Jim Utter	Sun, 07 Jun 202..
5edd7f2fc7980...	Veach stars on ...	https://cdn-1....	5ed917c51f462...	https://motors...	Zach Veach ma...	https://cdn-1....	David Malsher-...	Sun, 07 Jun 202..
5edd7f30c798...	Daly rejoices in...	https://cdn-1....	5ed917c51f462...	https://motors...	Conor Daly is t...	https://cdn-1....	David Malsher-...	Sun, 07 Jun 202..
5edd7f31c798...	NASCAR holds...	https://cdn-1....	5ed917c21f462...	https://motors...	NASCAR utilize...	https://cdn-1....	Nick DeGroot	Sun, 07 Jun 202..
5edd7f31c798...	NBC delivers b...	https://cdn-1....	5ed917c51f462...	https://motors...	The 2020 NTT I...	https://cdn-1....	David Malsher-...	Sun, 07 Jun 202..
5edd7f32c798...	Russell wins thi...	https://cdn-1....	5ed917b41f462...	https://motors...	George Russell ...	https://cdn-1....	Josh Suttill	Sun, 07 Jun 202..
5edd7f33c798...	Wolff says Ha...	https://cdn-1....	5ed917ab1f462...	https://motors...	Mercedes boss...	https://cdn-1....	Jonathan Noble	Sun, 07 Jun 202..
5edd7f34c798...	Vandoorne cro...	https://cdn-1....	5ed917b41f462...	https://motors...	Stoffel Vandoo...	https://cdn-1....	Matt Kew	Sun, 07 Jun 202..
5edd7f35c798...	Despite early s...	https://cdn-1....	5edca71799757...	https://motors...	Noah Gragson ...	https://cdn-1....	Jim Utter	Sun, 07 Jun 202..
5edd7f36c798...	Pirelli confiden...	https://cdn-1....	5ed917ab1f462...	https://motors...	Pirelli Formula ...	https://cdn-1....	Adam Cooper	Sun, 07 Jun 202..
5edd7f37c798...	What time and...	https://cdn-1....	5ed917c21f462...	https://motors...	NASCAR heads...	https://cdn-1....		Sun, 07 Jun 202..
5edd7f39c798...	Petrucci would...	https://cdn-1....	5eda022513003...	https://motors...	Outgoing fact...	https://cdn-1....	Lewis Duncan	Sun, 07 Jun 202..
5edd7f3ac798...	When a Superb...	https://cdn-1....	5eda022513003...	https://motors...	Called into repl...	https://cdn-1....	Lewis Duncan	Sun, 07 Jun 202..
5edd7f3bc798...	Alfa Romeo sq...	https://cdn-1....	5edca72199757...	https://motors...	The Romeo Fer...	https://cdn-1....	Rachit Thukral...	Sun, 07 Jun 202..
5edd7f3dc798...	What was behi...	https://cdn-1....	5ed917ab1f462...	https://motors...	While much of...	https://cdn-1....	Matt Somerfield	Sun, 07 Jun 202..
5edd7f3dc798...	Saudi Arabian ...	https://cdn-1....	5ed917b41f462...	https://motors...	Through SAFEl...	https://cdn-1....		Sun, 07 Jun 202..
5edd7f3ec798...	Aero handicap ...	https://cdn-1....	5ed917ab1f462...	https://motors...	Mercedes For...	https://cdn-1....	Adam Cooper	Sun, 07 Jun 202..
5edd7f3fc7980...	Dixon says Ho...	https://cdn-1....	5ed917c51f462...	https://motors...	Scott Dixon cre...	https://cdn-1....	David Malsher-...	Sun, 07 Jun 202..
5edd7f40c798...	Third-placed N...	https://cdn-1....	5ed917c51f462...	https://motors...	Defending NT...	https://cdn-1....	David Malsher-...	Sun, 07 Jun 202..
5edd7f40c798...	Rosenqvist acc...	https://cdn-1....	5ed917c51f462...	https://motors...	Felix Rosenqvis...	https://cdn-1....	David Malsher-...	Sun, 07 Jun 202..

Рис. 3.3. Колекція news.


calendar

_id	title	startDate	endDate	status	series
5edd7e8af76b...	Esports Formu...	2020-06-05T21:...	2020-06-05T21:...	Event finished	5edca7fa8c9aa...
5edd7e8af76b...	Esports Formu...	2020-06-06T21:...	2020-06-06T21:...	Event finished	5edd7e8af76ba...
5edd7e8af76b...	Formula 1 Aus...	1970-01-01T00:...	1970-01-01T00:...	25 days	5edca7fb8c9aa...
5edd7e8af76b...	Formula 1 Styri...	1970-01-01T00:...	1970-01-01T00:...	31 days	5edca7fb8c9aa...
5edd7e8af76b...	MotoGP Czech...	1970-01-01T00:...	1970-01-01T00:...	59 days	5edca7fb8c9aa...
5edd7e8af76b...	MotoGP Austri...	1970-01-01T00:...	1970-01-01T00:...	66 days	5edca7fb8c9aa...
5edd7e8af76b...	NASCAR Cup ...	2020-05-30T21:...	2020-05-30T21:...	Event finished	5edca7fb8c9aa...
5edd7e8af76b...	NASCAR Cup ...	2020-06-06T21:...	2020-06-06T21:...	Event finished	5edca7fb8c9aa...
5edd7e8af76b...	IndyCar Texas	2020-06-05T21:...	2020-06-05T21:...	Event finished	5edca7fb8c9aa...
5edd7e8af76b...	IndyCar Grand...	1970-01-01T00:...	1970-01-01T00:...	24 days	5edca7fb8c9aa...
5edd7e8af76b...	WEC Spa-Fran...	1970-01-01T00:...	1970-01-01T00:...	66 days	5edca7fb8c9aa...
5edd7e8af76b...	WEC Bahrain	1970-01-01T00:...	1970-01-01T00:...	164 days	5edca7fb8c9aa...

Рис. 3.4. Колекція calendar.

series

_id	name	url
5ed917ab1f46...	Formula 1	/Formula 1
5ed917ac1f46...	DTM	/DTM
5ed917af1f462...	NASCAR	/NASCAR
5ed917b21f46...	W Series	/W Series
5ed917b31f46...	IMSA	/IMSA
5ed917b41f46...	Esports	/Esports
5ed917b71f46...	BTCC	/BTCC
5ed917be1f46...	Super GT	/Super GT
5ed917c01f46...	WRC	/WRC
5ed917c11f46...	Supercars	/Supercars
5ed917c21f46...	NASCAR Cup	/NASCAR Cup
5ed917c31f46...	NHRA	/NHRA
5ed917c51f46...	IndyCar	/IndyCar
5eda02251300...	MotoGP	/MotoGP
5eda02bf4edd...	WEC	/WEC

Рис. 3.5. Колекція series.

Перш за все необхідно визначитися з дизайном сервісу, так він має досить важливу роль для користувача. Поганий дизайн може відштовхнути користувача, і людина не стане затримуватись на сайті. Інтерфейс повинен бути зрозумілим, не повинно бути постійно вспливаючої реклами або вікон, які будуть відволікати від користування. Також велике значення має адаптація під різні розміри екрану, так як користувачі будуть заходити з різних видів систем, як з комп'ютерів з широкоформатними моніторами, так і з мобільних приладів. Тому слід розробляти адаптивну верстку для того, щоб сервіс не втрачав функціоналу на різних пристроях і в той же час не втрачав зручності у користуванні.

Після визначення з дизайном, слід верстати головні компоненти. Для цього будемо використовувати мову React та css. Структура сервісу буде наступною:

- Header – шапка сайту, яка буде містити перехід на головну сторінку та згортаєму навігацію(фільтрацію);

- Content – тіло сайту, у якому будуть знаходитись або список новин(Container), або тіло новини(News) в залежності від обраної сторінки, а також календар найближчих подій(Calendar).

Для того, щоб полегшити життя розробникам, компанія Facebook, розробила модуль, який дозволяє створити шаблон React додатоку автоматично. Даний модуль називається create-react-app, який необхідно встановити глобально для того, щоб використовувати у будь-якому проекті. Далі достатньо перейти у потрібну папку, використовуючи командну строку, та ввести create-react-app. Прослідувавши інструкціїї отримуємо такий шаблон(Рис. 3.3):

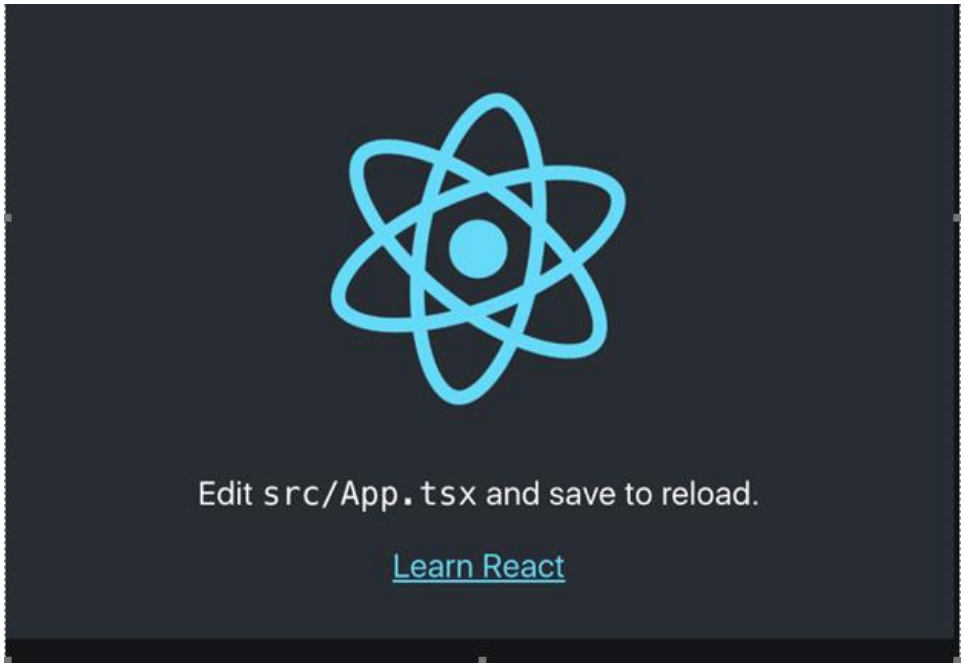


Рис. 3.3. Шаблон додатку create-react-app

Далі можна змінювати розмітку головного модуля App.js, створювати нові компоненти та працювати з вже налагодженим додатком.

Для більш легкого сприйняття коду, а також відображення різних компонентів в залежності від тих, чи інших умов слід розбивати великі компоненти на більш дрібні класи. Для цього достатньо створити папку з файлами .js(.jsx) і .css. Далі, у місті використання цього компоненту зробити import. Усі компоненти являють собою функції, які повертають розмітку jsx. У такі класи можна передавати будь-які параметри, але про них слід згадати пізні-

ше. Слід пам’ятати, що такі класи хоча і схожі на функції, але визиваються як звичайні елементи розмітки, наприклад <Header />. Ім’я класу обов’язково повинно починатися з великої літери, так як при використанні назв, які починаються з літер нижнього регістру, є велика ймовірність натрапити на зарезервовані команди чи елементи.

Компонент Header буде містити перехід на головну сторінку, кнопку згортання/розгортання навігації, та саму навігацію. Навігацію слід винести в окремий компонент, який буде називатися Sidebar. У навігації будуть виводитись усі серії, які є в базі даних, а саме їх назва, яка буде слугувати посиланням на відповідну сторінку. Кнопка «сховати» буде змінювати свій вигляд, в залежності від стану навігації(Рис. 3.4). Сама навігація буде розташована з лівого краю сторінки під Header-ом на комп’ютерній версії і по центру – на мобільній версії.

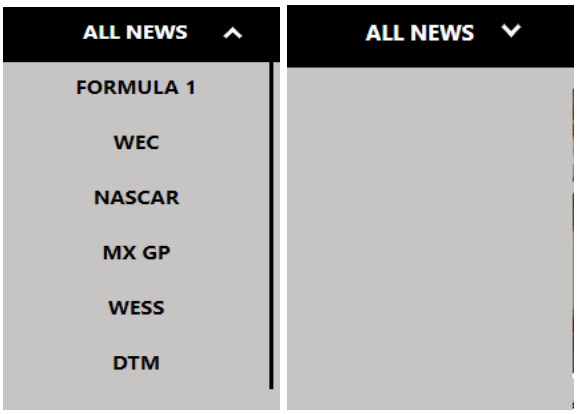


Рис. 3.4. Навігація. Зліва – розгорнутий стан, справа – згорнутий.

Далі створимо компонент Calendar. Він повинен відображати календар найближчих подій, їх стан, назву. Також слід додати функцію «згорнути», так як буде важко відобразити його на мобільних пристроях так, щоб не зменшити зручність використання. А так, з’являється можливість «розгорнути/згорнути» за необхідністю. Отже отримуємо компонент, який має заголовок, біля якого кнопка «згорнути/розгорнути». Під заголовком знаходяться елементи подій, які містять назву події, дати проведення, статус та серію(Рис. 3.5).

Рис. 3.5. Вигляд календара.

Переходимо до створення компоненту, яка відповідає за відображення усіх новин. У залежності від кількості новин, які виводяться, потрібно створити стільки ж компонентів. Для цього достатньо за допомогою функції map пройти по масиву новин і для кожної відобразити елемент з відповідними даними. Кожен елемент буде містити зменшене зображення, назву, короткий опис новини. Так як ми будемо виводити одий і той самий компонент для різних даних, слід винести його в окремий клас. В результаті отримуємо такий компонент(Рис. 3.6):

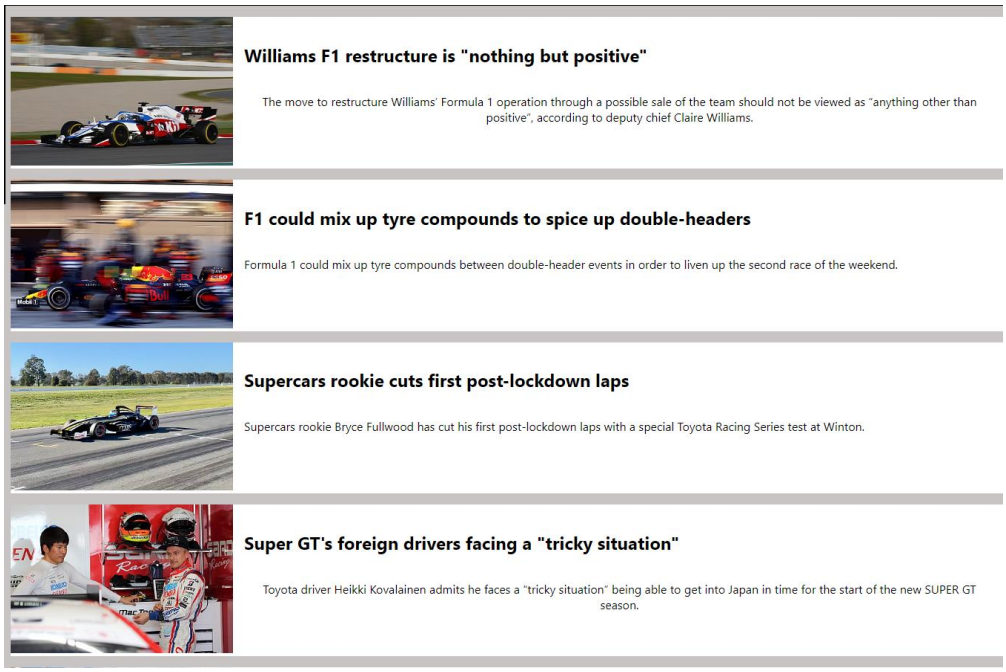


Рис. 3.6. Вигляд списку новин.

На останок залишилося розробити сторінку окремої новини. Вона буде складатися із зіголовку, зображення або галереї зображень, автора, дати викладення, підзаголовку та тексту новини. Навігацію та календар залишаємо на сторінці. Отримуємо такий вигляд(Рис. 3.7):

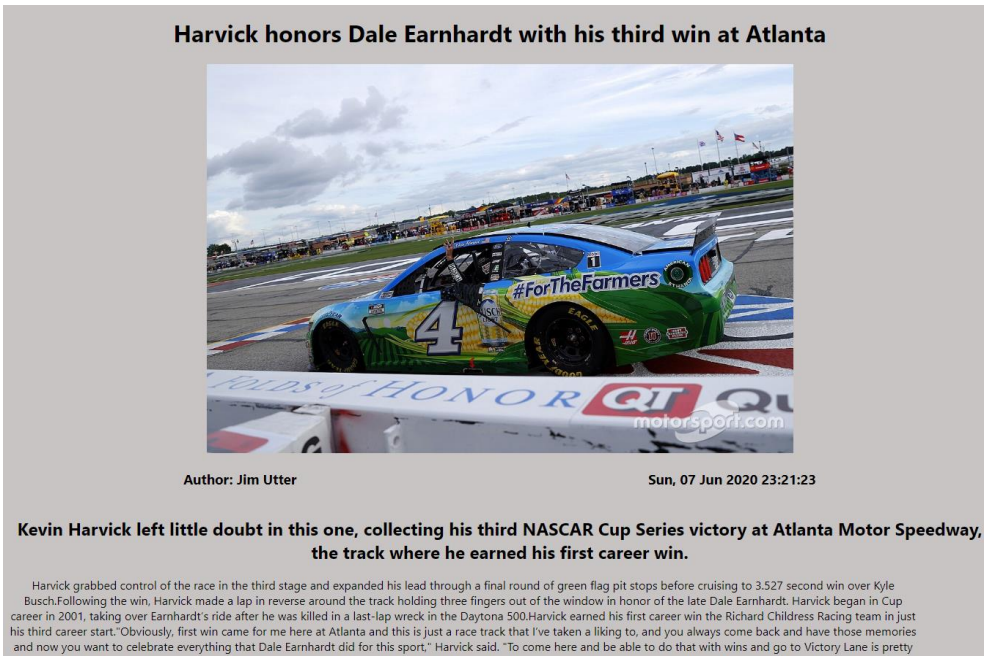


Рис. 3.7. Вигляд компоненту News.

Таким чином ми зверстали усі необхідні компоненти. На Рис. 3.8 та Рис. 3.9 зображені сторінки цілком.

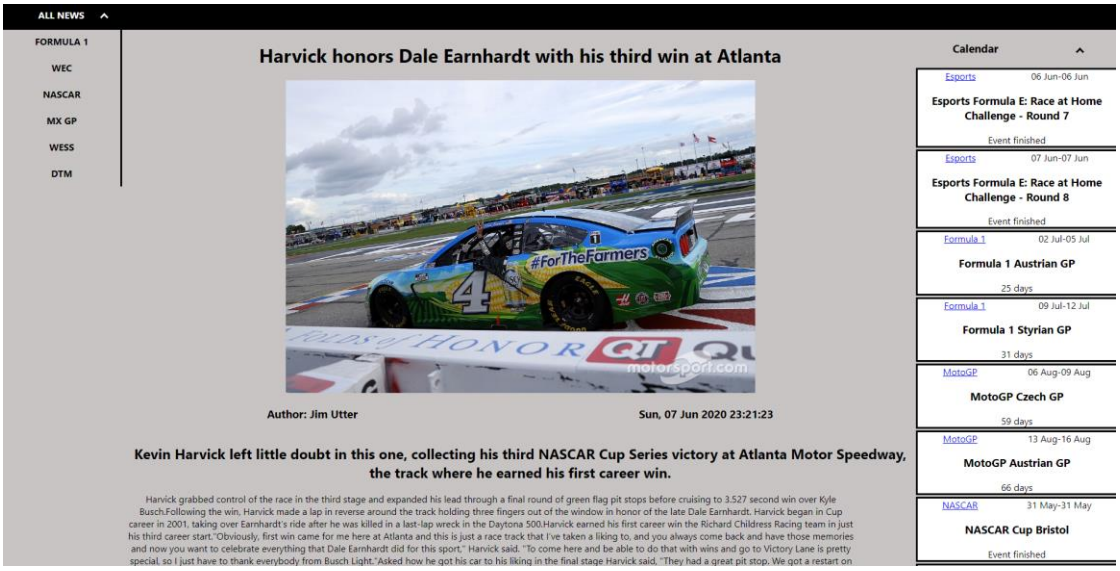


Рис. 3.8. Сторінка новини.

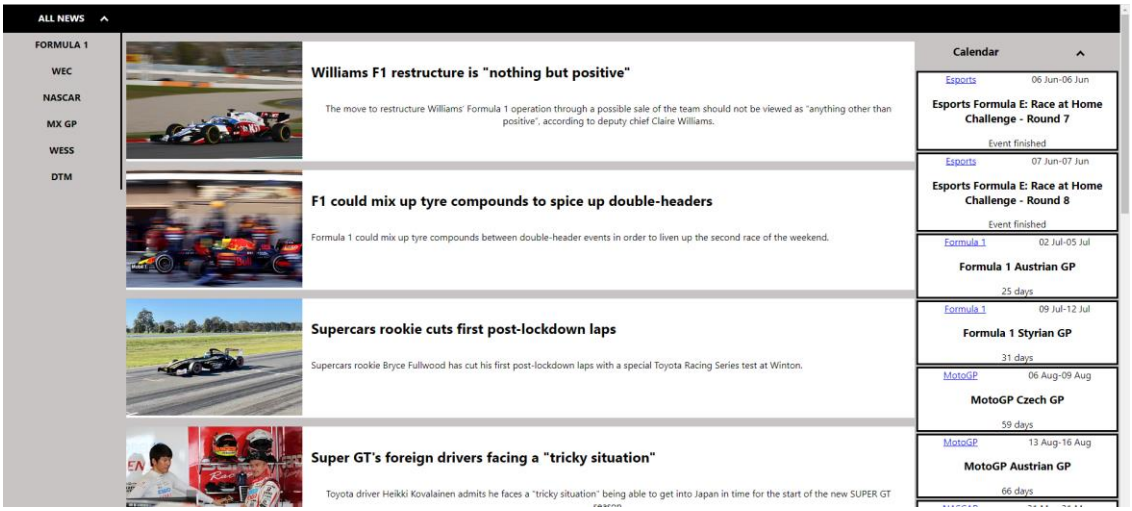


Рис. 3.9. Головна сторінка.

Далі слід зробити маршрутизацію. У будь-якому реальному веб-додатку потрібні маршрути, і додаток React не виняток. Користувач повинен бачити, де він знаходиться в додатку в будь-який момент часу. А бачить він

своє місце перебування в адресному рядку браузера. Отже додаток повинен вміти зіставляти певний URL з відповідною йому сторінкою. Тобто, якщо ми введемо в адресний рядок адресу сторінки, то програма має направити нас на конкретну сторінку, але не на будь-яку іншу.

Також повинна працювати історія. Тобто коли користувач клацає на стрілку "Назад" в браузері, програма має направити нас на попередню сторінку.

Сам по собі React не надає такої можливості, це завдання спеціальних бібліотек. Як правило, використовуючи API такої бібліотеки ми підключаємо компоненти сторінок нашого становища, зіставляючи їх з певними шляхами. Після цього, переходячи з однієї сторінки на іншу ми будемо бачити в адресному рядку, як змінюється поточний шлях.

На даний момент є кілька популярних бібліотек для маршрутизації: react-router, router5, aviator тощо. Повний список можна подивитися тут. Ми будемо використовувати react-router.

Ця бібліотека популярна, задоволена проста у використанні і має гарну документацією. Вона надає такі можливості як:

- Навігація по кліку (компонент <Link>);
- Перенаправлення (компонент <Redirect>);
- Маршрутизація (компонент Route);
- Історія (властивість history).

Зробивши структуру, зображену на Рис. 3.10, ми отримуємо можливість завантажувати лише той компонент, який відповідає введеному Url. Немає необхідності завантажувати усі компоненти одразу, а використовувати лише потрібний у даний момент.

```
function App() {

  return (
    <div className='App'>
      <Header />
      <div className = 'content'>
        <Switch>
          <Route path='/' component={News} />
          <Route path='/news' component={Container} />
        </Switch>
        <Calendar />
      </div>
    </div>
  );
}
```

Рис. 3.10. Структура модуля App.

Для того, щоб не перезавантажувати сторінку кожного разу, натискаючи на посилання, можна використовувати замість тегів <a/> тег <NavLink /> з бібліотеки react-router-dom. Він має атрибут to, в який можна передати посилання. Даний тег замысть перезавантаження всієї сторінки, змінює ті компоненти, які відносяться до даного Url. При компіляції тег <NavLink /> перетворюється на тег <a/>.

Залишилось підключити дані використовуючи створене API. Кожен клас завжди має деякі атрибути – props. Якщо не вказувати явно, то props набуває значення пустого об’єкту. За допомогою props ми будемо передавати дані у компоненти.

У компоненті календаря виконаємо запит на події, startDate яких більше або дорівнює дати сьогоднішнього дня, але викликаємо лише перші десять, відсортовані по даті. Таким чином ми відсіяли непотрібні події та зменшили час запиту шляхом отримання лише десять документів, а не всіх. Будемо передавати отримані дані у props елемента для використання. Так ми отримали назву, дату, статус та серію події, передаємо їх у відповідні

елементи. У компоненті усіх новин будемо робити запит усіх новин або конкретної серії, якщо вона обрана і передавати дані у props компонентів. Так як відображати усі новини одночасно досить складно і затратно з точки зору ресурсів, розбиваємо новини на сторінки. Для цього після запиту отримуємо кількість усіх отриманих документів та ділимо на бажану кількість новин на сторінці. Тоді на одній сторінці виводимо лише певну кількість новин. Зі списком серій виконуємо аналогічні дії.

Висновок до Розділу 3

Отже, у даному розділі були порівняні різні технології для розробки бекенд частини, фронтенд частини, а також проаналізовані дві популярні бази даних з виявленням переваг і недолік кожної. Був зроблений та аргументований вибір мов та інструментів для розробки кожної з частин. Також у даному розділі наведені усі етапи розробки веб-сервісу з детальним оглядом застосованих технологій та прийомів для виконання поставлених задач. Також продемонстровані результати виконання кожного з етапів.

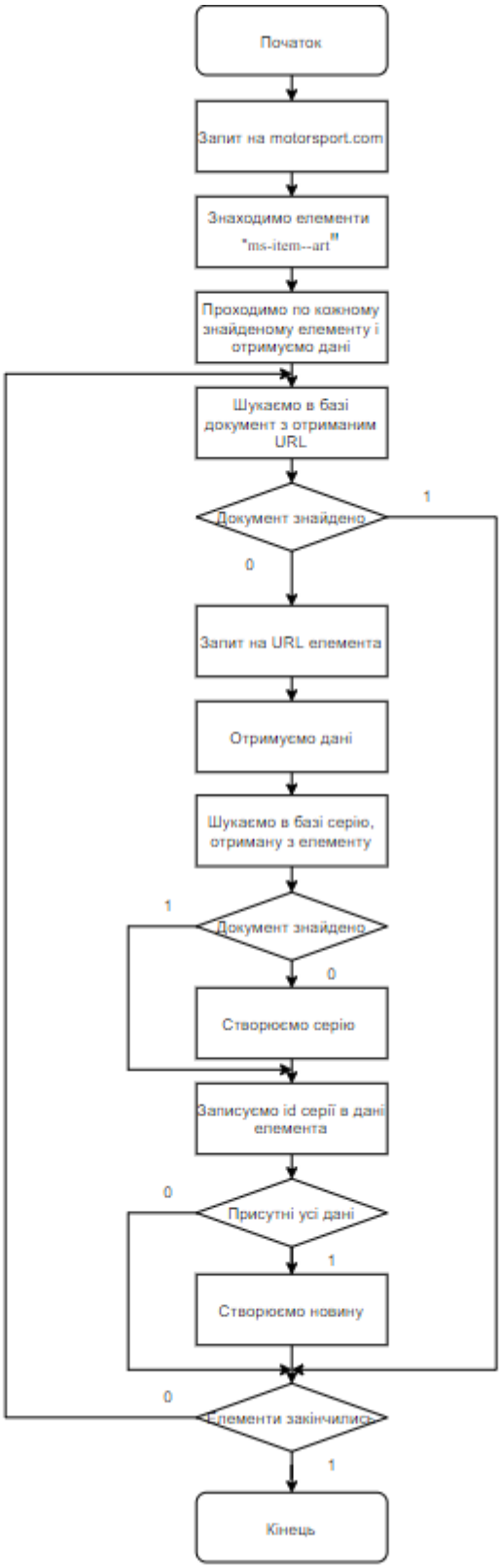
ВИСНОВКИ

Під час виконання даної бакалаврської роботи були поглиблені знання, а також закріплені навички проектування веб-додатків. У рамках цього проекту були досягнуті такі результати:

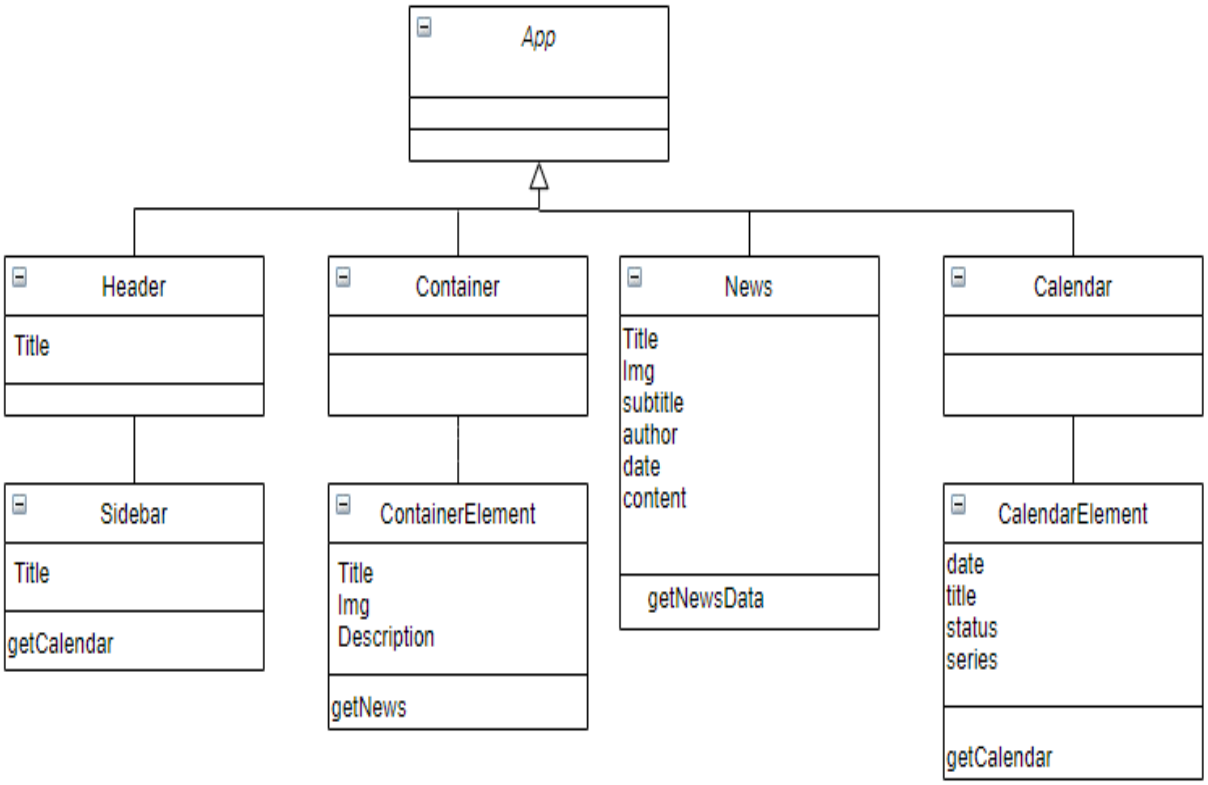
1. Наведені теоритичні відомості про різні види інтернет-ресурсів, їх призначення, їх архітектура та їх класифікація. Окремо розглянуті веб-служби, з усіма існуючими їх підтипами та короткою характеристикою про кожен з них. Були проаналізован переваги та недоліки різних типів.
2. Були порівняні різні платформи та інструменти для розробки різних складових веб-сервісу. Був проведений аналіз двох головних видів баз даних на прикладі MySQL та MongoDb, виділені їх відмінності, переваги та недоліки. Проаналізовані засоби розробки дали змогу визначити, які інструменти слід використовувати для виконання поставленої задачі та поглибити знання з даної теми.
3. Був проведений аналіз інструментів для розробки різних складових веб-сервісу, на основі якого був зроблений вибір засобів. Спроектований та розроблений веб-сервіс, який відповвідає поставленим вимогам і наведені результати розробки кожного з етапів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

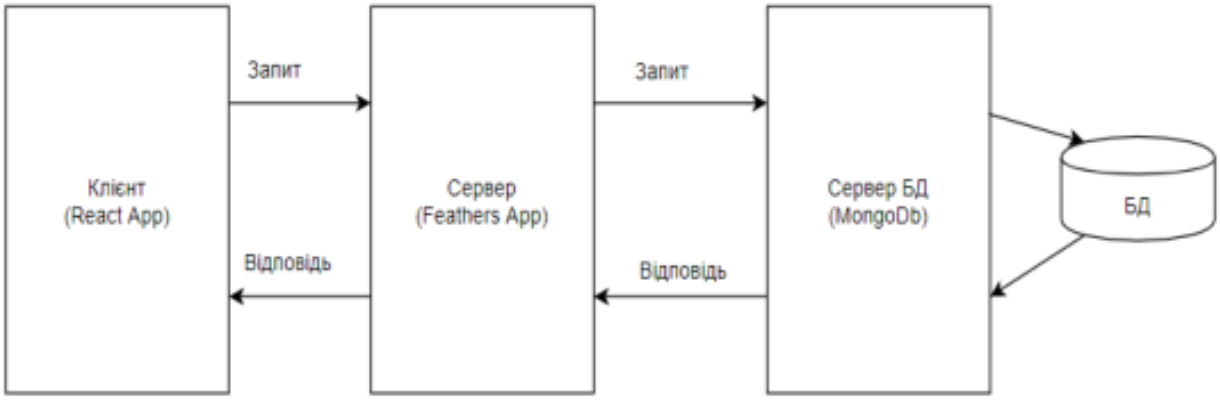
1. Feathers JS Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.feathersjs.com/.
2. Современный учебник JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://learn.javascript.ru/.
3. Полное руководство по React [Електронний ресурс] – Режим доступу до ресурсу: https://learn-reactjs.ru/.
4. Intro to React [Електронний ресурс] – Режим доступу до ресурсу: https://reactjs.org/.
5. React Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://devdocs.io/react/.
6. Npmjs | Build amazing things [Електронний ресурс] – Режим доступу до ресурсу: https://npmjs.com/.
7. Документация | Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/.
8. HTML | MDN [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/HTML.
9. CSS | MDN [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/CSS.
10. React или Angular или Vue.js — что выбрать? [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/476312/.
11. MongoDB vs MySQL: A Comparative Study on Databases [Електронний ресурс] – Режим доступу до ресурсу: https://www.simform.com/mongodb-vs-mysql-databases/.
12. Python или Node.Js: какой язык выбрать? [Електронний ресурс] – Режим доступу до ресурсу: https://nuancesprog.ru/p/4281/.
13. MySQL и MongoDB — когда и что лучше использовать [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/322532/.



					ІАЛЦ.467200.005 Д1		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Дем'яненко М.О.*			**Веб-сервіс з автоспортивними новинами**	*Літ.*	*Аркуш*	*Аркушів*
Перев.		*Верба О.А..*					*1*	**1**
Тех.контр.					**Схема програми**	***НТУУ "КПІ ім. Ігоря Сікорського", ФІОТ,ОТ, ІО-61***		
Н.Контр.		*Сімоненко В.П..*						
Затвердж.								



					ІАЛЦ.467200.006 Д2		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Дем'яненко М.О.*			**Веб-сервіс з автоспортивними новинами**	*Літ.*	*Аркуш*	*Аркушів*
Перев.		*Верба О.А..*					*1*	**1**
Тех.контр.					**Діаграма класів**	***НТУУ "КПІ ім. Ігоря Сікорського", ФІОТ,ОТ, ІО-61***		
Н.Контр.		*Сімоненко В.П..*						
Затвердж.								



					ІАЛЦ.467100.007 ДЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Дем'яненко М.О.			Веб-сервіс з автоспортивними новинами **Схема взаємодії програми з користувачем**	Літ.	Аркуш 1 / Аркушів 1
Перев.		Верба О.А..					
Тех.контр.						НТУУ "КПІ ім. Ігоря Сікорського", ФІОТ,ОТ, ІО-61	
Н.Контр.		Сімоненко В.П..					
Затвердж.							